IN THE SPECIFICATION:

Applicants hereby amend the following paragraphs of the specification as noted. These amendments correct typographical errors and add no new matter.

**[0005]** One problem associated with motion estimation is that it is an expensive computing activity in the encoding process, i.e., typically associated with calculation intensive steps. For example, in FIG. 9, for each match, exhaustively searching a (64 x 64 pixels) search area [[1]]222 for a (16 x 16 pixels) matchblock [[1]]220, involves undertaking: 256 comparisons (differences); 256 calculations involving computing the absolute values of the differences (referenced subsequently as "absolute value calculations" for convenience and brevity) between the matchblock and a search area; and 255 additions of the 256 values. For 64 matches, the total computations involved increases drastically to 64•64•256 + 64•64•256 + 64•64+255. Because motion estimation conventionally requires an exhaustive search of a large area of the matchblock, it can thus be appreciated that a need exists for a technique to reduce the number of computations required to determine the motion vector information. It is desirable to reduce the large number of calculations traditionally involved to determine the motion vector.

**[0056]** In FIG. 2, a general block diagram is provided for the DCT/IDCT and quantization and dequantization functionality. Those skilled in the art will recognize that these functions may be provided in a variety of ways. For example, the output of the discrete cosine transform module 42 is quantized. An output of quantizer 46 can be reconstructed through inverse quantizer 42, and provided through inverse DCT 42 for storage in memory, like macro-block SRAM 60 so that it may undergo decompensation. The general purpose of subjecting the output of quantizer 46 through inverse quantization and inverse DCT 42 is to determine a difference macroblock that is lossy. When data is summed with the output of the MEC engine 38 along with dedicated object-based motion prediction tools (not shown) for each object, a lossy version

of the original picture can be stored in SRAMs 4[[4]]1 or 43. Those skilled in the art will appreciate that motion texture coding, DCT-based texture coding (e.g., using either standard 8 x 8 DCT or shape-adaptive DCT) may also be provided with front end 1[[1]]3. It is noted that DCT/IDCT module 42 and quantizer/dequantizer module 46 not only provide the above functions during the encoding process, but [[is]] are capable of emulating a decoder for the reconstruction of frames based on receiving the reference frames along with the motion vectors. To this end, the functional blocks 38-48 are capable of operating as a codec.

**[0058]** Those skilled in the art will recognize that the blocks of FIG. [[1]] 2 are functional blocks that may be implemented either by hardware, software, or a combination of both. Given the functional description of these blocks, those of ordinary skill in the art will be able to implement various components described using well-known combinational and/or sequential logic, as well as software without undue experimentation. Those skilled in the art will appreciate that the present invention is not limited to the video compression system described above, but is applicable to any video processing system.

**[0063]** Referring to FIG. 4, one embodiment of the MEC array 66 is shown. The MEC cell array 66 is a processor cell array that includes 7 x 22 = 154 cells 68, a portion of which comprises a sub-array 50 (shown in FIG. [[2]] 3) having 5 x 20 = 100 cells, each used for processing (i.e., major computing) and referred to as p_cells 80. MEC array 66 further includes 4 types of boundary cells, namely corner cells 82, column cells 84, updown cells 86 (5 up, 5 down), and io cells 88, which are generally used to store data for use by various components of the MEC engine 38. The interconnections of the p_cells 80 include 4 inputs and 4 outputs so that data can be transferred to each p_cell 80 from a neighboring p_cell 80 in the array 66. One aspect of the present invention is to reduce the complexity of the routing, placement, and layout

of logic of the MEC cell array 66. With the embodiment of FIG. 4, this is achieved because each p_cell 80 of the 5 x 20 sub-array 50 is connected to its neighboring p_cell 80. This simplifies the logic design complexity to providing adjacent interconnections amongst p_cells 80. With a simple layout, placement and routing of logic, manufacturing costs of system 10 are reduced, which is particularly beneficial for SoC and ASIC applications.

**[0075]** In performing the step of partial pixel level searching 102, searches on a smaller picture size (i.e., subsampled pixels) are performed to determine a preliminary motion vector ($mv_{QxQ}$) 232. In particular, an M x M pixel-sized matchblock 220 (e.g., macroblock) of a current frame is decomposed 112 (i.e., subsampled) into a smaller size picture, denoted by a sub-matchblock 226 of size Q x Q, where Q< M. At times, reference to Q x Q will be made to $Q^2$, primarily for convenience. For example, if M = 16, and Q = 4, then a 16 x 16 block is split into four 4 x 4 sub-matchblocks to find the preliminary motion vector $mv_{QxQ}$ 232 for each 4 x 4 block. This effectively decomposes the original picture into smaller sub-sampled pictures from which to interpolate data. For example, if the current frame is 640 x 480 pixels, then a $Q^2$ size search can be selected to be 4 x 4, thereby resulting in a block of 160 x 120 pixels. Each pixel of the $Q^2$ size picture is the average of every 16 pixels forming the original picture. With substep 112, the sub-matchblock will be 4 x 4, instead of 16 x 16. Similarly, the search area 222 of the reference frame 224 is decomposed 114 by a factor of Q into search area sub-blocks 228, so that an exhaustive $Q^2$ search can be performed in the (modified) search area sub-blocks 228. In substep 116, the $Q^2$ size (4 x 4) matchblock data 226 is loaded 116 into the cell array 66, and distributed, that is duplicated to 16 identical ones across each sub-block 228, as seen in FIG. [[9]] 10A.

**[0076]** Referring to FIG. 9, a matchblock 220 is typically scanned over a corresponding search area 222 within a reference frame 224 to find the best match. This process is repeated for each matchblock in the search area 222. By contrast, when performing the $Q^2$ search of the present invention, several sub-tasks may be performed in parallel. For example, referring back to FIG. 10A, the matchblock 220 has been split (subsampled) into Q x Q sub-matchblocks 226. Similarly, the search area 222 has been split into 4 x 4 sub-blocks 228, each of 16 x 16 pixels. Using the MEC cell array 66 of the present invention, within a (e.g., 16 x 16) search area sub-block 228, the Q x Q pixel sub-matchblock 226 can be duplicated as each sub-matchblock 226 is of size 4 x 4. This duplication can be repeated 16 times over the search area sub-blocks 228. Each of the 16 sub-matchblocks 226 can then be searched in parallel using reduced size (e.g., 7 x 7) sub-search areas [[1]] 230. Also in substep 116, the $Q^2$ size reference area is loaded into cell array 66. By way of example, the search is performed on a 4x4 sub-matchblock 226 in a 19x19 area, which is selected as a reasonably large sub-search area. Accordingly, a total of 16x16 matches are needed. Amongst the 16 sub-matchblocks 226, each being of 4x4 pixels, for each sub-matchblock 226, only (16x16)/16 = 16 = 4x4 matches are needed. For a QxQ = 4x4 sub-matchblock in a 4x4 search, the search area will be 7x7. For B-type frames, the search area 222 is typically 73x73 pixels for previous and post reference frames, while with P-type frames, the search area is typically 153x73 pixels for left and right parts of the reference frame.

**[0080]** One manner of implementing this transfer is with a calling procedure to the RISC CPU 18 coordinated by controller 20. Thereafter, substep 130 is undertaken to move the matchblock relative to the refined search area 240 and so that continuing SAD calculations can be performed as part of the exhaustive search in the full size (20 x 20) level search. In FIG. 10B, a distance [[1]] 244 of $\pm 2$ around the pixel location 245 indicated by $mv_{QxQ}$ 232 is a 5 x 5 pixel sub-array 242. Accordingly, a total computation of 5 x 5 = 25 matches are undertaken, where

each match has 16 x 16 = 256 calculations performed as part of the exhaustive search. Once these steps are performed, the best match intermediate motion vector $mv_{full}$ 246 can be found [[1]] 232, indicating how many pixels in the y direction and how many pixels in the x direction is the best match displaced in the current picture relative to the reference picture. It will become apparent to those skilled in the art that with the substeps 124-132, the full size search attempts to improve the motion estimation of step 102, that is, by determining a full pixel level motion vector, $mv_{full}$ 246, which is a refinement of the $mv_{QxQ}$ 232. By way of example, the preliminary motion vector $mv_{QxQ}$ = (4, 8) might be further refined to (5,6) with step 104.

**[0082]** For example, referring to FIG. 10A, for one sub-matchblock 226 in a search area sub-block 228, there is a 4:1 sub-sampled representation. Because such search area sub-block 228 is a 16 x 16 pixel area in which to search for sub-matchblock 226, which is 4 x 4 pixels in size, the exhaustive search for one match involves undertaking 16 comparisons (differences); 16 calculations involving computing the absolute value of the differences; and 15 additions of the 16 values. So, for (16x16 =) 256 matches in the search area [[1]] 222, the total computations involved is 256•16 comparisons + 256•16 absolute values + 256•15 additions. This is a significant reduction in the number of calculations by comparison to conventional methods of using the matchblock to exhaustively search the entire search area. When the full pixel level search is undertaken to refine the granularity of the preliminary motion vector $mv_{QxQ}$ 232, the computation will be based on 25 matches. So, even though there are additional stages associated with refining the motion vector in accordance with the present invention, the overall number of calculations is significantly reduced.